Multi-threading the generation of Burrows-Wheeler Alignment

H. Jo

Department of Information Technology, Chonbuk National University, Republic of Korea

Corresponding author: H. Jo
E-mail: heeseung@jbnu.ac.kr

Genet. Mol. Res. 15 (2): gmr.15028650
Received March 23, 2016
Accepted April 11, 2016
Published May 23, 2016
DOI http://dx.doi.org/10.4238/gmr.15028650

ABSTRACT. Along with recent progress in next-generation sequencing technology, it has become easier to process larger amounts of genome sequencing data at a lower cost. The most time-consuming step of next-generation sequencing data analysis involves the mapping of read data into a reference genome. Although the Burrows-Wheeler Alignment (BWA) tool is one of the most widely used open-source software tools for aligning read sequences, it still has a limitation in that it does not fully support a multi-thread mechanism during the alignment generation step. In this article, we propose a BWA-MT tool based on BWA that supports multi-thread mechanisms for processing alignment generation. To evaluate BWA-MT, we used an evaluation system equipped with 24 cores and 128 GB of memory. As workloads, we used the hg19 human genome reference sequence and sequences of various read sizes from the 1 to 40 M spots. In our evaluation, BWA-MT showed a maximum of 3.66-times better performance, and generated the same Sequence Alignment/Map result file as that of BWA. Although the ability to speed up the procedure might be dependent on computing resources, we confirmed that BWA-MT is a highly effective and fast alignment tool.

Key words: Genome sequencing; Next-generation sequencing; Burrow-Wheeler Aligner tool; Multi-thread

INTRODUCTION

Since the Human Genome Project was successfully completed, genome sequencing technology has been rapidly developing. Owing to the evolution and progress in next-generation sequencing (NGS) technology, it now takes only one week to analyze the whole genome of an individual, and data generation has become faster at a lower cost compared to the traditional methods. This production of explosive genome data has introduced a new paradigm shift for research in the life sciences.

One of the most crucial steps of NGS data analysis is the mapping of read data into a reference genome. However, this alignment analysis is the most laborious part of all of the analysis steps, in terms of time and computing resource consumption. To cope with these drawbacks of NGS alignment analysis, various alignment software tools have been developed, such as Burrows-Wheeler Alignment (BWA) (Li and Durbin, 2009), MAQ (Li et al., 2008a), BOWTIE (Langmead et al., 2009), and SOAP (Li et al., 2008b). All of these tools use indexing mechanisms to minimize the time required for mapping, and are largely classified into hash table-based indexing and suffix tree-based indexing methods. Among them, BWA, BOWTIE, and SOAP2 allow for high-speed alignment based on the Burrows-Wheeler Transformation (BWT) algorithm (Burrows and Wheeler, 1994); it is well known that BWT achieves the highest mapping quality and performance.

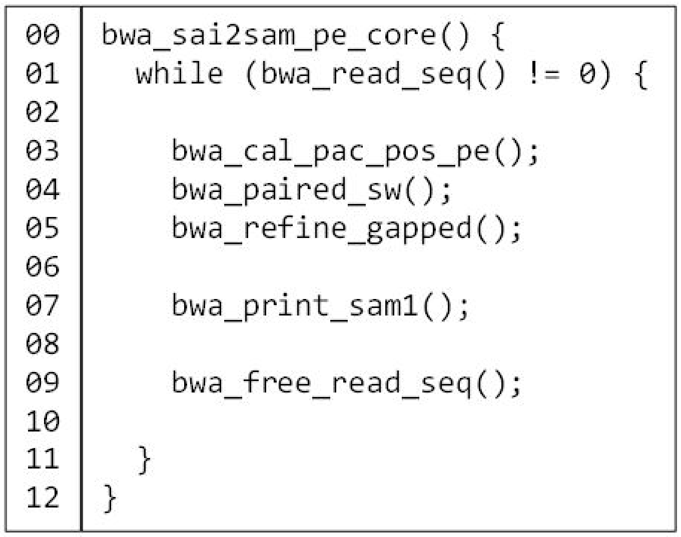
Currently, BWA is an open-source tool and is one of the most commonly used tools in whole-genome read alignment analysis. The alignment process of BWA is divided into two parts: 1) mapping (BWA aln) and 2) alignment generation (BWA samse/sampe). In step (1), BWA finds suffix array (SA) coordinates for each of the read sequences using the pre-built BWT index and the SA of a reference genome sequence. This step is implemented for the use of multi-threading and takes a relatively short time for processing. Based on the information of step (1), the generated Suffix Array Index (SAI) file should be translated into the actual genomic coordinates, and BWA generates the output into the Sequence Alignment/Map (SAM) file format via step (2). Unfortunately, because this step does not support multi-threading, it cannot fully utilize the underlying computation resources such as multi-core processors, and therefore takes a substantial amount of time.

To resolve this issue, we implemented and improved BWA-MT, based on BWA and BWA-MT (single-ends version; Jo and Koh, 2015), in a manner that supports the multi-thread computation of paired-ends for step (2). With incorporation of multi-threading, BWA-MT can fully utilize multi-core processors and substantially shortens the time required to generate alignment data. Although the performance enhancement is dependent on the computing resources and sequence data size, BWA-MT showed up to 3.66-times better performance in our evaluation and generated the same SAM result file compared with BWA.

Design and implementation

BWA-MT is a modified version of the sampe step of BWA to support multi-threading. Our approach was to split a read sequence into several parts and process each part separately on each thread. This approach is similar to the processing mechanism of BigData frameworks such as MapReduce (Dean and Ghemawat, 2008) and Hadoop. In general, it divides a whole input data stream into many chunks and allocates jobs to process each chunk using multi-core processors and multiple nodes in parallel. Owing to this parallelization mechanism, the BigData frameworks can achieve high machine utilization and performance throughput. BWA-MT splits a long read sequence given as an input into several partitions according to the number of threads set by a user, and allocates each of these threads accordingly so that each partition is processed independently and concurrently.

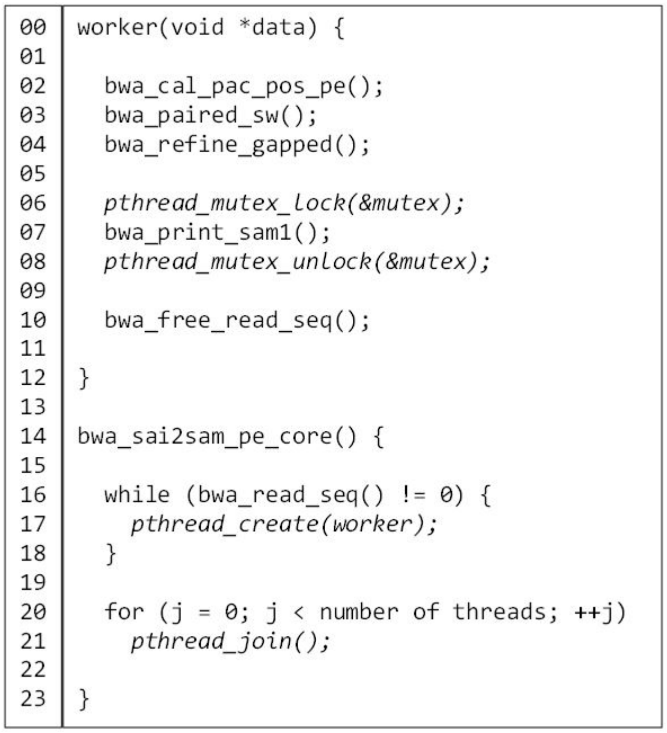
Figures 1 and 2 depict the comparison of core alignment generation procedures between BWA and BWA-MT. The *bwa_sai2sam_pe_core* is a core function that generates alignment in BWA. First, it reads a certain amount of the alignment (01) and converts it into sequence coordinates by labeling it the *bwa_cal_pac_pos_pe* function (03). Then, it refines gapped alignments (04-05) and finally converts the result into the SAM file format (09). These processing steps are reiterated until all input read sequences are processed. Unfortunately, this algorithm and implementation are designed for use with only a single processor core. Alternatively, in the algorithm of BWA-MT, the *bwa_sai2sam_pe_core* function creates threads named *worker*. The number of threads and the size of a read sequence subset are pre-determined by the total size of the read sequence, and each thread is launched with a data structure to support each read sequence subset. The *worker* function defines the function of a thread and performs similar computation to that of BWA.



```
00  bwa_sai2sam_pe_core() {
01    while (bwa_read_seq() != 0) {
02
03      bwa_cal_pac_pos_pe();
04      bwa_paired_sw();
05      bwa_refine_gapped();
06
07      bwa_print_sam1();
08
09      bwa_free_read_seq();
10
11    }
12  }
```

Figure 1. Core alignment generation procedure of BWA.

When processing is carried out in this manner, there are three considerations related to the implementation of BWA that require discussion. The first is accessing the index of BWT. Since the alignment generation consults the BWT index of a reference sequence, BWA-MT reads ahead before launching each thread and then shares the information among threads. The second consideration is that the computation results must be converted into the SAM file format. Since this process should be ordered, BWA-MT holds each of the results in memory during the computation of each thread and then serializes them into the SAM file after finishing the computation of each thread. The last consideration concerns the number of threads analyzed. BWA-MT sets the number of threads automatically, and therefore a user does not need to configure the number of threads. BWA-MT is designed to process 262,144 spots by a thread, which is similar to the BWA workflow, which processes 262,144 spots at once. Therefore, the total number of threads is decided according to the total number of spots. For example, if a read sequence includes 524,288 spots, BWA-MT launches two threads automatically.



```
worker(void *data) {

  bwa_cal_pac_pos_pe();
  bwa_paired_sw();
  bwa_refine_gapped();

  pthread_mutex_lock(&mutex);
  bwa_print_sam1();
  pthread_mutex_unlock(&mutex);

  bwa_free_read_seq();

}

bwa_sai2sam_pe_core() {

  while (bwa_read_seq() != 0) {
    pthread_create(worker);
  }

  for (j = 0; j < number of threads; ++j)
    pthread_join();

}
```

Figure 2. Multi-threaded alignment generation procedure of BWA-MT.

Evaluation

To evaluate the performance of BWA-MT, we used an evaluation system equipped with 24 cores and 128 GB of memory. The version of BWA used was 0.7.5a, and the proposed BWA-MT was also modified from this same version. Table 1 shows the reference and read sequences that were used in our evaluations. The read sequences were retrieved from SRR1580822 (http://www.ncbi.nlm.nih.gov/sra/SRX699055 [accn]), and we increased the number of spots.

Table 1. The workloads used in the evaluations.

Dataset name	Number of spots	Size (Mb)
Ref. (hg19)	6,177,547,390 (num. of char.)	3,083
1M	1,000,000	672
10M	10,000,000	6,766
20M	20,000,000	13,574
30M	30,000,000	20,383
40M	40,000,000	27,192

Execution time

Figure 3 shows the execution time results of BWA and BWA-MT. The y-axis denotes the execution time and time required for the sampe step, and the x-axis shows the read sequences listed in Table 1. The length of each read sequence was 100 bp and was a paired-end read. As shown in Figure 3, the performance enhancement of BWA-MT improved by up to 3.66 times compared to that of BWA. As the length of the read sequence increased, the performance gap also increased, since the computation part of the sampe step becomes larger, and the multi-threaded processing of BWA-MT can thus contribute to performance enhancement.

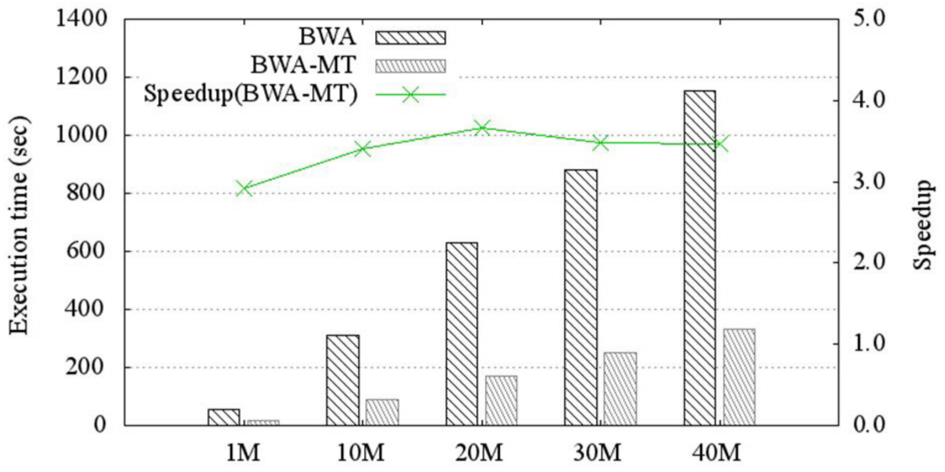


Figure 3. Alignment generation time of BWA and BWA-MT for each of the read sequences. The green line denotes the speedup of BWA-MT compared to BWA.

CPU utilization and memory consumption

Figure 4 shows the CPU and memory usages when implementing BWA and BWA-MT. The performance gap between BWA and BWA-MT was clearly evident in terms of CPU utilization. The average CPU utilization of BWA was 4%, whereas that of BWA-MT was 15%. This means that the multi-threading mechanism of BWA-MT can utilize more of the CPU cores. To compare the memory consumption between BWA and BWA-MT, we measured the peak memory usage. BWA-MT used more memory than BWA by a maximum of 45%. The space overhead is caused by the multi-thread mechanism, which holds the computation results of each thread in memory until all of the threads have been processed. It is unavoidable to serialize the output SAM file. Note that the described memory overhead reflects the peak usage of the system, including the page cache of Linux.

Confidence

In each workload run, we compared the difference of the SAM file results between BWA and BWA-MT. We confirmed that the two SAM file results were exactly same, which means that BWA-MT has the same alignment confidence and error rates as BWA.

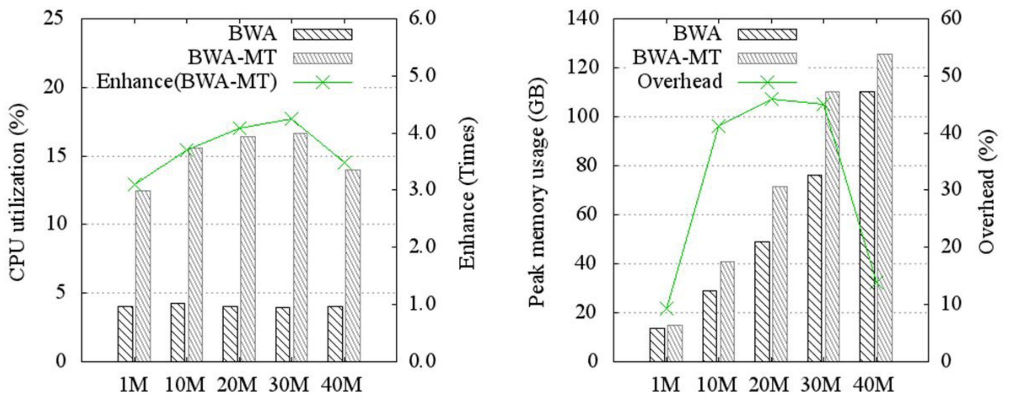


Figure 4. CPU utilization and peak memory usage of BWA and BWA-MT for each of the read sequences. The green line denotes the speedup and the overhead percentage of BWA-MT compared to BWA.

DISCUSSION

In this paper, we propose a new BWA-MT tool, based on BWA, that supports multi-thread computation. It was designed to fully utilize the underlying multi-core architecture of the computing resources. By using multi-threading, BWA-MT can substantially shorten the time required to generate an alignment for single-end or paired-end read sequences. Moreover, BWA-MT generates the same SAM result file as obtained with BWA. In the future, we plan to carry out further optimization, including the described SAM file serialization.

ACKNOWLEDGMENTS

Research supported by the Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (#B0101-15-0155, the Core Technology Development of SW-SoC Convergence Platform for Hyper-Connection Services among Smart Devices based on Heterogeneous Multi-core Clusters), the ICT R&D program of MSIP/IITP (#B0101-15-0644, the research project on High Performance and Scalable Manycore Operating System), and research funds of Chonbuk National University.

REFERENCES

Burrows M and Wheeler DJ (1994). A block-sorting lossless data compression algorithm. Technical Report 124, Palo Alto, CA, Digital Equipment Corporation.

Dean J and Ghemawat S (2008). MapReduce: simplified data processing on large clusters. *Commun. ACM* 51: 107-113. http://dx.doi.org/10.1145/1327452.1327492

Jo H and Koh G (2015). Faster single-end alignment generation utilizing multi-thread for BWA. *Biomed. Mater. Eng.* 26 (Suppl 1): S1791-S1796. http://dx.doi.org/10.3233/BME-151480

Langmead B, Schatz MC, Lin J, Pop M, et al. (2009). Searching for SNPs with cloud computing. *Genome Biol.* 10: R134.1-10. http://dx.doi.org/10.1186/gb-2009-10-11-r134

Li H and Durbin R (2009). Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics* 25: 1754-1760. http://dx.doi.org/10.1093/bioinformatics/btp324

Li H, Ruan J and Durbin R (2008a). Mapping short DNA sequencing reads and calling variants using mapping quality scores. *Genome Res.* 18: 1851-1858. http://dx.doi.org/10.1101/gr.078212.108

Li R, Li Y, Kristiansen K and Wang J (2008b). SOAP: short oligonucleotide alignment program. *Bioinformatics* 24: 713-714. http://dx.doi.org/10.1093/bioinformatics/btn025